```
 subclass s=new subclass();
 s.m1();
 }
}
```

Exception Handling

An exception is an event, which occurs during the execution of a program that dirupt the normal flow of the program's instructions.

There are two types of exception :

i) pre-defined or built-in exception -> These usually indicate programming bugs, such as logic errors or proper use of an API. For example, division by zero problem.

Some built-in exceptions are :

a) ClassNotFoundException
b) CloneNotSupportedException
c) InterruptedException
d) NoSuchFieldException
e) NoSuchMethodException
f) ArithmeticException
g) ArrayStoreException
h) NumberFormatException
i) NullPointerException

Exception handling keywords

1) try -> The first step in constructing an exception handler is to enclose the code that might throw an exception within a try block.

Syntax

```
try
  {
  code
  }
```

2) catch -> The block of the statements which are to be executed when an exception occurs must be written in catch block.

Every try block should contain atleast one catch block.

If an exception occurs within the try block, that exception is handled by an exception handler associated with it. To associate an exception handler with a try block, we must put a catch block after it.

A single try block can also contain multiple catch blocks.

Syntax :

```
try
  {
  …….
  }
catch(ExceptionType name)
```

```
        {
        …..
         }
       catch(ExceptionType name)
        {
        ….
       }
```

3) finally-> The finally block always executes when the try block exits. This ensures that the finally block is executed even if an unexpected exception occurs.

```
finally
  {
  code
   }
```

4) throw -> All methods use the throw statement to throw an exception. The throw statement requires a single argument : a throwable object. Throwable objects are instances of any subclass of the throwable class.

Syntax : throw somethrowableobject;

5) throws-> If a method is capable of causing an exception that it doesn't handle, it must specify the throws behaviour.

The throws cause lists the types of exceptions that a method might throw. This throws cause is :

```
return type mname(parameter list) throws ExceptionList
    {
    ……..
     }
```

Example :

```
import java.io.*;
class example
  {
   public static void main(String args[])
     {
      try
      {
       int a=5,b=0;
       double c=a/b;
       System.out.println(c);
       }
      catch(ArithmeticException ae)
        {
        System.out.println(ae);
        }
     }
  }
```

User-defined Exceptions

A programmer can also his own set of exceptions. The advantage of creating a user-defined exception class is as follows :

i) It can hold more information about the error condition than a standard class. This becomes specially more important when the error condition is more complicated.

Eg,

```
 import java.io.*;
 class RangeException extends RuntimeException
     {
      public void disp()
           {
            System.out.println("The given no. not in 1 to 10");
            }  }
 class example
      {
      public static void main(String args[])
          {
           try
            {
        int x;
        DataInputStream dis=new DataInputStream(System.in);
        System.out.println("Enter a no. ");
        x=Integer.parseInt(dis.readLine());
        if(x<1 || x>10)
                throw new RangeException();
        else
                System.out.println("number accepted");
        }
        catch(IOException ioe)
           {
            System.out.println(ioe);
            }
         catch(RangeException re)
           {
           re.disp();
           }
        finally
           {
           System.out.println("I am from finally block");
           }
}
}
```

Applet programming

Applets are small java programs that are primarily used in Internet computing. They can be transported over the Internet from one computer to another and run using the appletviewer or any web browser that supports java.

Difference between applet and application

i) Applet don't use the main() method for initiating the execution of the code. When loaded automatically call certain methods of applet class to start and execute the applet code.

ii) Unlike stand-alone applications, applets can't be run independently. They are run from inside a web page using a special feature known as HTML tag.

iii) Applets can't read from or write to the files in the local computer.

iv) Applets are restricted from using libraries from other languages such as C or C++.

The steps involved in developing and testing in applet are :

1. Building an applet code (.java file)
2. Creating an executable applet (.class file)
3. Designing a web page using HTML Tags.
4. preparing <APPLET> tag.
5. Incorporating <APPLET> tag into the web page.
6. Creating HTML file.
7. Testing the applet code.

Write a simple program in Applet to print "Hello Java"

```
import java.awt.*;
import java.applet.*;
public class app1 extends Applet
  {
  public void paint(Graphics g)
    {
    g.drawString("Hello java",40,60);
    }
  }
```

Compile -> javac app1.java

abc.html

```
<html>
<applet code="app1.class" width=400 height=400>
</applet>
</html>
```